

Further Studies on Heuristics

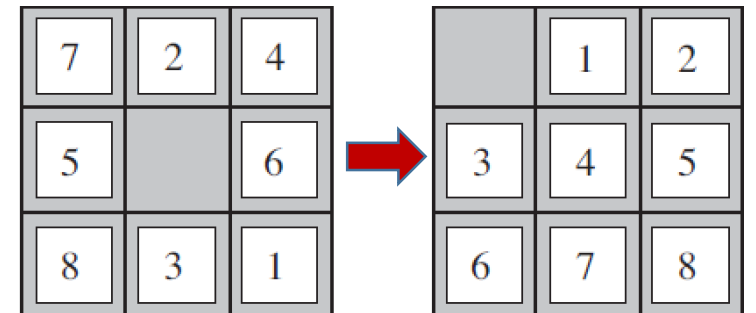
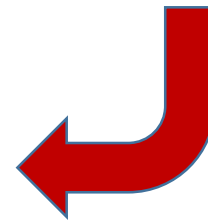
I. Search Efficiency of Heuristics

Recall: Heuristics for 8-puzzle

- $h_{mis}(s) = \# \text{misplaced tiles} \in [0,8]$: **Admissible**.
- $h_{1stp}(s) = \#(1\text{-step move})$ to reach the goal configuration: **Admissible**.

➤ $h_{1stp}(s) \geq h_{mis}(s) \Rightarrow h_{1stp}(s)$ is '**better**' than $h_{mis}(s)$.

What does 'better' mean?



Dominance

- For **admissible** h_1 and h_2 , if $h_1(s) \geq h_2(s)$ for $\forall s$
 $\Rightarrow h_1$ **dominates** h_2 and is **more efficient** for search.

- **Theorem**: For any admissible heuristics h_1 and h_2 , define

$$h(s) = \max\{h_1(s), h_2(s)\}$$

$h(s)$ is admissible and dominates both h_1 and h_2 .

- **‘Better’ heuristic = dominance = better search efficiency.**

Even Better Dominance

- **Question:** Which one to choose from a collection of admissible heuristics h_1, \dots, h_m & none dominates any other?
- **Answer:** $h(s) = \max\{h_1(s), \dots, h_m(s)\}$ dominates all the others.

Quantify Search Efficiency

- **Effective Branching Factor b^*** : 通过实验得到 For a solution from A*, calculate b^* satisfying:
$$N = b^* + (b^*)^2 + \dots + (b^*)^d$$
 - N : #nodes of the solution,
 - d : depth of the solution tree.
 - E.g., A* finds a solution at depth 5 using 52 nodes $\Rightarrow b^* = 1.92$.
- Good heuristics have b^* close to 1 \Rightarrow large problems solved at reasonable computational cost.
- **b^* quantifies search efficiency of heuristics.**

Empirical: Factor b^*

- **Aim**: Compare h_1 and h_2 regarding the search efficiency.
- **Setting**: Generate 1200 random problems with $d = \{2, \dots, 24\}$ and solve them with IDS and A* with h_1 & h_2 .
- **Note**: IDS – a baseline.

$$\begin{aligned} & \{n \mid f(n) \leq C^*\} \\ & \Downarrow f(n) = g(n) + h(n) \\ & \{n \mid h(n) \leq C^* - g(n)\} \end{aligned}$$

$g(n)$ 与 $h(n)$ 无关

$\rightarrow \{n \mid h_1(n) \leq C^* - g(n)\}$

包含

$\rightarrow \{n \mid h_2(n) \leq C^* - g(n)\}$

搜索空间更小.
better

Empirical: Factor b^*

b^*

	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Empirical: Factor b^*

	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4						
6						
8						
10						
12						
14						
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

- h_2 is 'better' than h_1 regarding search efficiency.
- This goodness is reflected by b^* being closer to 1.
- A* with h_2 performs much better than IDS.

II. Generate Admissible Heuristics

We Know about Heuristics ...

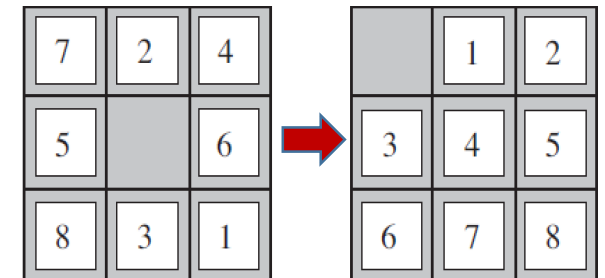
- We know:
 - How to judge their admissibility.
 - How to compare their goodness regarding searching efficiency.
- **Question:** How to produce such 'good' heuristics?

(1) Generate from Relaxed Problems

Where are h_{mis} & h_{1stp} from?

For 8-puzzle problem:

- **Real Rule:** A tile can only move to the **adjacent empty** square.
- **Relaxed rules:** h_{mis} and h_{1stp} are admissible
 - R1: A tile can move **anywhere** $\Rightarrow h_{mis}(s) = \#(\text{misplaced tiles})$.
 - R2: A tile can move one step in **any direction** regardless of an occupied neighbour $\Rightarrow h_{1stp}(s) = \#(1\text{-step move})$ to reach goal.
- **Optimal solutions to problems with R1, R2 are easier to find.**



Relaxed Problem

- **Relaxed problem**: a problem with **relaxed rules** on the action.
- E.g. 8-puzzle problems with R1 and R2. 放松规则后的问题
- **Theorem**: The cost of an optimal solution to **a relaxed problem** is an **admissible heuristic** for the original problem.
- No wonder h_{mis} and h_{1stp} are admissible.

(2) Generate from Sub-problems

Subproblem 子问题

- Subproblem 只考虑其中4个子

- Task: get tiles 1, 2, 3 and 4 into their correct positions.

- Relaxation: move them disregarding the others.

- Theory: $\text{cost}^*(\text{subproblem}) < \text{cost}^*(\text{original})$.

- $\text{cost}^*(\text{subproblem})$: the cost of the optimal solution of this subproblem.

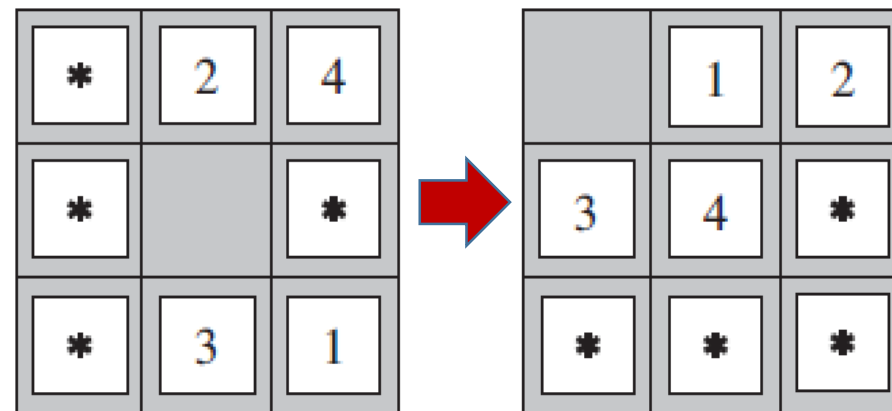


Fig.1. A subproblem of 8-puzzle.

Subproblem and Admissible Heuristics

- **Admissible** $h_{sub}^*(s)$: estimate the cost from s to the subproblem goal.
 - E.g. $h_{sub}^{(1,2,3,4)}$ is the cost to solve the 1-2-3-4 subproblem.
- **Theorem**: $h_{sub}(s)$ dominates $h_{1stp}(s)$,
 - $h_{sub}(s) = \max\{h_{sub}^{(1,2,3,4)}(s), h_{sub}^{(2,3,4,5)}(s), \dots\}$.

Disjoint Subproblems

- **Question:** Will the **addition of heuristics** from subproblem (1-2-3-4) and (5-6-7-8) give an **admissible heuristic**, considering the two subproblems are not overlapped?
- **Answer:** No, since they always **share some moves**.
- **Question:** What if **not count** those shared moves?
- **Answer:** $h_{sub}^{(1,2,3,4)}(s) + h_{sub}^{(5,6,7,8)}(s) \leq c^*(s) \Rightarrow$ admissible.
 - Disjoint pattern database

(3) Generate from Experiences

‘Experience’ Formulation

For 8-puzzle problem:

- Solve many 8-puzzles to obtain **many examples**.
- Each **example** consists of a state from the solution path and the actual cost of the solution from that point.
- These **examples** are our ‘**experience**’ for this problem.
- **Question**: How to learn $h(s)$ from these **experience**?

ML
向量 → 线性回归

Learn Heuristics from Experience

- **Question:** What are the **good experience features**?
- **Answer:** **Relevant** to predicting the states' cost to Goal, e.g.
 - $x_1(s)$: #(displaced tiles).
 - $x_2(s)$: #(pairs of adjacent tiles) that are not adjacent in Goal state.
- **Question:** How to learn h from those **relevant experience features**?
- **Answer:** (e.g.) Construct model as

不能保证 add missive

$$h(s) = w_1 x_1(s) + w_2 x_2(s), +b$$

where w_1, w_2 are model parameters to learn from training data by a learning method such as neural networks and decision trees.